EPtask: Deep Reinforcement Learning based Energy-efficient and Priority-aware Task Scheduling for Dynamic Vehicular Edge Computing

Peisong Li, *Student Member, IEEE,* Ziren Xiao, *Student Member, IEEE,* Xinheng Wang, *Senior Member, IEEE,* Kaizhu Huang, *Senior Member, IEEE,* Yi Huang, *Fellow, IEEE,* Honghao Gao, *Senior Member, IEEE*

***Abstract*—The increasing complexity of vehicles has led to a growing demand for in-vehicle services that rely on multiple sensors. In the Vehicular Edge Computing (VEC) paradigm, energy-efficient task scheduling is critical to achieving optimal completion time and energy consumption. Although extensive research has been conducted in this field, challenges remain in meeting the requirements of time-sensitive services and adapting to dynamic traffic environments. In this context, a novel algorithm called Multi-action and Environment-adaptive Proximal Policy Optimization algorithm (MEPPO) is designed based on the conventional PPO algorithm and then a joint task scheduling and resource allocation method is proposed based on the designed MEPPO algorithm. In specific, the method involves three core aspects. Firstly, task scheduling strategy is designed to generate task offloading decisions and priority assignment decisions for the tasks utilizing PPO algorithm, which can further reduce the completion time of service requests. Secondly, transmit power allocation scheme is designed considering the expected transmission distance among vehicles and edge servers, which can minimize transmission energy consumption by adjusting the allocated transmit power dynamically. Thirdly, the proposed MEPPO-based scheduling method can make scheduling decisions for vehicles with different numbers of tasks by manipulating the state space of the PPO algorithm, which makes the proposed method be adaptive to real-world dynamic VEC environment. At last, the effectiveness of the proposed method is demonstrated through extensive simulation and on-site experiments.**

***Index Terms*—Proximal Policy Optimization, task scheduling, resource allocation, vehicular edge computing.**

This work was supported in part by the Key Program Special Fund in XJTLU under project KSF-E-64, in part by the XJTLU Research Development Funding under projects RDF-19-01-14 and RDF-20-01-15, and in part by the National Natural Science Foundation of China (NSFC) under grant 52175030. (*Corresponding authors: Xinheng Wang and Honghao Gao.*)

P. Li, Z. Xiao and X. Wang are with the School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China, E-mail: peisong.li20@student.xjtlu.edu.cn; ziren.xiao20@student.xjtlu.edu.cn; xinheng.wang@xjtlu.edu.cn

K. Huang is with the Data Science Research Center & Division of Natural and Applied Sciences, Duke Kunshan University, Suzhou 215316, China, E-mail: kaizhu.huang@dukekunshan.edu.cn

Y. Huang is with the Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool L69 3BX, UK, E-mail: yi.huang@liverpool.ac.uk

H. Gao is with the School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China, E-mail: gaohonghao@shu.edu.cn

I. Introduction

A. Background

Nowadays, vehicles are equipped with advanced features and rely on multiple onboard sensors to provide intelligent in-vehicle services [1]. These services encompass a wide range of functionalities, ranging from safety features such as automatic emergency braking and blind-spot monitoring to entertainment systems offering high-quality audio and video streaming [2]. The increasing demand for intelligent in-vehicle services highlights the significance of developing efficient data processing methods based on multiple sensors, making it a highly promising area of research [3]. When it comes to in-vehicle services, completion time and energy consumption of services are two crucial factors. Minimizing completion time is particularly important for safety-critical services such as navigation and autonomous driving because it reduces data processing delays. In addition, vehicles have limited power resources, and multi-sensor in-vehicle services require substantial computational power, leading to significant energy consumption [4]. By optimizing in-vehicle computation to reduce energy consumption, the overall battery life of the vehicle can be extended, which is especially crucial for electric vehicles.

In order to achieve the goal of minimizing completion time and reducing energy consumption, the Vehicular Edge Computing (VEC) paradigm offers a solution [5]. VEC decomposes multiple sensors-based service requests into multiple computation tasks, which can be distributed among various computing entities such as the cloud server, edge servers, and other vehicles. Processing these tasks in parallel helps to decrease the time required to complete the requests. Moreover, this approach offloads the computation tasks to more powerful computing entities, thus reducing energy consumption on the vehicle's onboard resources [6]. Generating optimal task offloading and resource allocation decisions are two important ways for VEC to enhance the performance of multi-sensor in-vehicle services. Therefore, extensive research has been conducted on task offloading and resource allocation in VEC in recent years, the latest studies are reviewed in Section II in this paper.

Despite the extensive research on task offloading and resource allocation to improve completion time and energy consumption, there remains a challenge in meeting the requirements of time-sensitive services. The completion time of a service request is crucial in delivering satisfactory service to end users. Therefore, it is essential to further enhance task offloading and resource allocation algorithms to meet the demands of time-sensitive services while minimizing energy consumption.

Deep Reinforcement Learning (DRL) is attracting attention due to its suitability for decision-making problems where an agent interacts with an environment [7]. Proximal Policy Optimization (PPO) is a policy gradient algorithm proposed in 2017 [8]. As a refinement to Trust Region Policy Optimization (TRPO) (2015) [9], PPO uses a simpler clipped surrogate objective, omitting the expensive second-order optimization presented in TRPO. In 2018, PPO2 was introduced, which offers better GPU utilization by batching observations from multiple simulation environments. Presently, PPO has become the default reinforcement learning algorithm at OpenAI due to its good performance. PPO incorporates a mechanism for controlling the update step size, helping ensure stable and incremental policy updates. This feature makes PPO less sensitive to large policy changes and improves the stability of the learning process. In the context of task scheduling, this stability is beneficial because it allows the scheduling policy to be updated gradually and avoids abrupt changes that could negatively impact the system's performance.

B. Motivation and Contributions

In order to further reduce request completion time and energy usage as well as adapt to the dynamic VEC scenario, the following problems are considered and corresponding solving methods are proposed:

In VEC, priority refers to the relative importance of a task compared to other tasks and it determines the order in which tasks are executed. If tasks are not scheduled based on their priority, it can lead to longer processing time and resource waste. If the task is assigned with a fixed priority, the overall performance can also be negatively influenced because of the dynamic environment. Therefore, in this work, the processing priority is assigned to each task dynamically to determine the execution order. Then, tasks allocated to the same computing entity are executed in the order of priority.

In addition to the processing priority, the adjustment of transmit power, one of the communication resources, is another crucial aspect of VEC. It involves adjusting the transmit power of the wireless communication devices used by vehicles and edge servers to optimize energy consumption and reduce interference. Current studies typically use a fixed transmit power. However, if the transmit power is too high, it can lead to unnecessary energy consumption and interfere with other devices operating in the same frequency band. In contrast, if the transmit power is too low, the range of communication can be limited, leading to communication failure when the devices are far apart. Starting from 2023, there has been a focus on optimizing transmit power allocation to minimize transmission energy consumption [10]–[12]. Recent research has explored the use of reinforcement learning algorithms to optimize transmit power allocation, where the allocation of transmit power is considered as one of the actions in the reinforcement learning process. However, the challenge lies in the continuous nature of transmit power, making the learning process quite complex [13]. Although discretizing the action space can simplify the learning process, it becomes difficult to achieve the optimal action since the optimal power allocation might fall between the discretized actions. Therefore, in this work, the transmit power is adjusted dynamically to reduce the transmission cost while ensuring transmission reliability.

Moreover, current DRL-based studies are only capable of allocating either a single task or a fixed number of subtasks for the vehicle [14]. However, in real-world scenarios, the number of vehicles generating service requests within the communication range of an edge server is constantly changing. Additionally, the number of tasks in the request generated by each vehicle varies due to different service requirements and the collaboration of different onboard sensors. This paradigm can be named the dynamic Vehicular Edge Computing paradigm. Therefore, the task scheduling method must be adaptive and able to make scheduling decisions for vehicles with various service requests comprising different numbers of tasks. Therefore, a task scheduling method is designed in this paper that can adapt to different numbers of tasks in a dynamic VEC paradigm, considering the ever-changing number of vehicles. The designed method can make decisions, including task offloading, communication resource allocation, and priority assignment, for a different number of tasks.

In summary, in order to further reduce the delay and adapt to real-world dynamic traffic volume, a Multi-action and Environment-adaptive Proximal Policy Optimization algorithm (MEPPO) is proposed in this paper and then a joint task scheduling and resource allocation method is designed based on the proposed MEPPO algorithm. In specific, firstly, the designed method optimizes the request's completion time by dynamically offloading tasks and assigning a priority to each task based on the task offloading decisions and priority assignment decisions. Secondly, the energy consumption can be optimized by adjusting the transmit power based on the inter-vehicle and vehicle-to-server distances. Thirdly, the size of the generated decisions is adjusted automatically according to the dynamic traffic environment, in which the number of vehicles and service requests is dynamic.

To the best of the authors' knowledge, this paper is the first work that studies how to perform task scheduling for various numbers of tasks. This is also the first work proposing to dynamically assign priority to the task.

The main contributions are summarised as follows:

(1) A joint task scheduling and resource allocation method

is designed. The task scheduling in this work is defined including task offloading and priority assignment. The importance of task priority in determining the order of task execution and avoiding longer processing time and resource waste is first recognized. This dynamic priority assignment ensures optimal task execution order and contributes to further reducing request completion time.

(2) A dynamic transmit power adjustment approach is proposed that optimizes the transmission cost while ensuring reliable communication. By dynamically adjusting the transmit power based on the specific requirements and transmission distances involved, the proposed method contributes to improving the overall performance of in-vehicle services in terms of energy consumption and communication reliability.

(3) A multi-action and Environment-adaptive Proximal Policy Optimization algorithm (MEPPO) is proposed, which is adaptive and capable of making scheduling decisions for vehicles with varying service requests comprising different numbers of tasks in the dynamic VEC paradigm. The designed algorithm encompasses decisions related to task offloading and priority assignment, accommodating different numbers of tasks efficiently. By addressing the dynamic nature of the VEC paradigm and developing an adaptive task scheduling method, the scalability and flexibility of multi-sensor in-vehicle services can be enhanced.

(4) The proposed method is evaluated by both simulations and on-site tests. The on-site test scenario is designed to represent the real-world environment. Experimental results reveal that the method can improve the performance of multi-sensor services within the dynamic VEC paradigm.

C. Paper Organization

The rest of this paper is organized as follows: In Section II, related work is reviewed. It also demonstrates how our work is different from existing research. In Section III, the system model is introduced. The proposed MEPPO-based joint task scheduling and resource allocation method is introduced in Section IV. The performance evaluation and results are presented in Section V. The on-site test is presented in Section VI. Finally, this paper is concluded in Section VII. The key abbreviations are shown in Table I.

TABLE I: List of main abbreviations

Abbrevation	Description
VEC	Vehicular Edge Computing
MEC	Mobile Edge Computing
V2V	Vehicle to Vehicle
V2I	Vehicle to Infrastructure
E2C	Edge server to Cloud server
V2E	Vehicle to Edge server
PPO	Proximal Policy Optimization
SAC	Soft Actor Critic
DQN	Deep Q-Network
GBD	Generalized Benders Decomposition
DDPG	Deep Deterministic Policy Gradient
A3C	Asynchronous Advantage Actor Critic
MAD4PG	multi-agent distributed distributional deep deterministic policy gradient

II. Related work

In this section, existing studies on reinforcement learning-based task scheduling and resource allocation are reviewed.

A. Studies on Task Scheduling

Firstly, much research focuses on task scheduling [22], [23]. In [24], an intelligent task offloading scheme was proposed based on deep Q learning to address the issue of the insufficient computing capacity of intelligent connected vehicles in VEC networks, which integrates mobile edge computing (MEC) and vehicular networks. In [25], a Markov decision process model was proposed to solve the computation offloading scheduling problem using proximal policy optimization algorithm in VEC scenarios, where the trade-off between task latency and energy consumption is minimized by considering both the location and timing of task execution. In [26], an energy-efficient task offloading and scheduling algorithm was proposed, which offloads delay-sensitive tasks to mobile fog vehicles instead of remote clouds to meet the computational demands of smart vehicles near rural highways. They presented a fuzzy reinforcement learning algorithm combined with a greedy heuristic algorithm to address the challenges of energy consumption optimization and scheduling decision-making with task deadline and resource availability constraints. In [27], a two-stage machine learning-based vehicular edge orchestrator was designed to efficiently operate a VEC system in the highly dynamic environment of autonomous vehicles, where computationally intensive workloads are offloaded to a nearby VEC infrastructure. The proposed approach considers task completion rate and service time to make a crucial decision about where to offload each task.

Based on the reinforcement learning algorithms, the task scheduling strategy can dynamically generate task scheduling decisions to reduce task completion time. However, in order to reduce completion time, most tasks are offloaded to edge servers, which could cause high energy consumption for edge servers.

B. Joint Task Scheduling and Resource Allocation

In order to solve the above-mentioned problem of high energy consumption, many studies focused on the joint optimization of completion time and energy consumption via designing a joint task offloading and resource allocation strategy. In [16], a solution for the joint task offloading scheduling and resource allocation problem was proposed by formulating it as a mixed integer optimization problem. They use a two-layer optimization approach where a Deep Q-Network (DQN) was used in the upper layer to solve the task offloading scheduling problem, and Gradient Descent was used in the lower level for CPU frequency allocation. In [17], a joint task offloading and resource allocation scheme was proposed to minimize the total task processing delay of all vehicles. In [18], a joint task offloading and resource allocation scheme was proposed for a parked-and-moving-vehicles-assisted Multi-access Edge Computing scenario, aiming to minimize the total priority-weighted task processing delay for all the devices. In [13], a MEC-enabled vehicular network was proposed to assist through aerial-terrestrial connectivity using high-altitude platforms

TABLE II: Comparison with the latest related studies

Work	Year	Method	Task scheduling		Resource allocation			Dynamic vehicles and tasks	Optimization target	
			Task offloading	Priority assignment	Conputation resource	Bandwidth	Transmit power		Delay	Energy
[15]	2021	SAC	Y						Y	Y
[16]	2022	DQN	Y		Y				Y	Y
[17]	2022	GBD	Y		Y	Y			Y	
[18]	2022	GBD	Y		Y	Y			Y	
[13]	2022	DQN	Y		Y	Y	Y			Y
[10]	2023	MAD4PG	Y		Y		Y		Y	
[11]	2023	DQN			Y	Y	Y		Y	
[19]	2023	A3C	Y		Y				Y	
[12]	2023	DDPG	Y				Y		Y	
[20]	2023	DDPG	Y			Y			Y	
[21]	2023	SAC	Y			Y			Y	
Proposed		PPO	Y	Y			Y	Y	Y	Y

equipped with mobile servers to provide computation offloading capability and network access for vehicle-to-vehicle communications.

By considering the allocation of computation and bandwidth resources, thus a balance between completion time and energy consumption can be achieved. However, above studies did not consider the allocation of transmit power, which is critical for reducing energy consumption when transmitting tasks. From this year (2023), transmit power, as one of the important communication resources, was started to be considered when designing resource allocation methods. In the latest published paper [28], a joint computation offloading and transmission resource allocation method was proposed, which combined non-orthogonal multiple access (NOMA) and multi-access edge computing. In summary, current studies solved the problem of transmit power allocation either by setting it as one of the actions in the reinforcement learning method or converting it into a convex optimization problem [10], [11], [13]. However, if the transmit power takes the continuous value, continuous action spaces can be highly complex and it's difficult for the DRL to learn optimal policies. If the transmit power is converted into discrete values, discrete actions can be inefficient in representing actions that require continuous adjustments. Discretizing a continuous action space may impose unrealistic constraints and limit the agent's performance. In addition, optimizing the transmit power as an independent sub-problem will result in a high algorithm complexity.

C. Dynamic VEC Paradigm

Current work focused on the allocation of either one task or a fixed number of sub-tasks for the vehicle. In [20], the number of moving vehicles on the road was fixed, and each vehicle generated a task to be implemented. In [19], it was assumed that each vehicle generated at most one task at each time slot. The task generation events were independently and identically distributed at each time slot. In [14], each vehicle needed to divide the task into multiple sub-tasks, and then distribute them to different vehicles for parallel execution. There is no dependency between the two sub-tasks and all tasks were split into the same-size sub-tasks.

These studies assumed each vehicle generated a fixed number of tasks or sub-tasks at a time slot. However, for the actual scenario, the number of vehicles and the number of tasks generated from the vehicle depends on the service provided by the vehicle at that moment. Current work cannot make adaptive scheduling decisions for the vehicle.

A more comprehensive analysis of the latest studies is shown in Table II, in which "Y" denotes yes, representing that the factor is considered in the paper. As shown in the table, from 2022, researchers started to focus on the joint decision making of task offloading and resource allocation, with the optimization target of minimizing task completion delay. However, the priority assignment has never been explored in previous studies. In addition, previous studies only considered task scheduling for traffic scenarios with a fixed number of vehicles and tasks, which is not feasible for real-world scenarios with a dynamic number of vehicles. In this paper, a joint task scheduling and resource allocation method is proposed to reduce both request completion time and energy consumption. The method can generate optimal task offloading decisions, priority assignment decisions, and transmit power allocation decisions simultaneously for the vehicles under the changing traffic volume.

III. System Model and Problem Definition

In this section, the dynamic VEC scenario is presented.

A. Network Model

The VEC system is illustrated in Fig. 1, which involves the use of computing resources in close proximity to vehicles to enable faster and more efficient processing of data. The paradigm includes three main components: vehicles, edge servers, and the cloud server. Vehicles can be divided into two categories: Task Vehicles that continuously generate computation tasks while moving and Service Vehicles that receive offloaded tasks. The tasks can be executed either locally in the vehicle, transmitted wirelessly to a nearby edge server or another vehicle in its vicinity, or transmitted to the

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2023.3321679

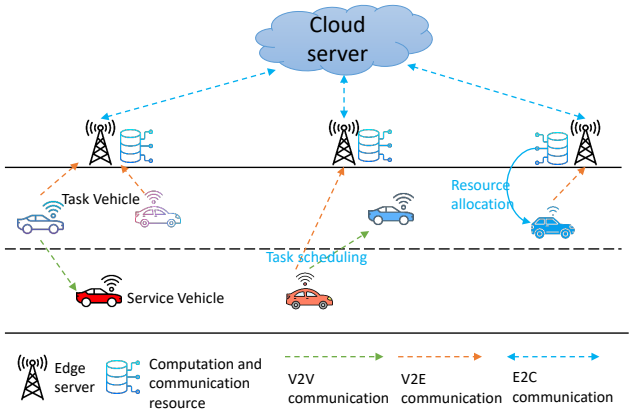


Fig. 1: Vehicular edge computing paradigm

cloud server. The edge servers usually consist of roadside units, base stations, or other traffic infrastructure, which can perform real-time analysis of the data generated by vehicles, enabling faster decision-making and reducing the latency of communication. Cloud server possesses sufficient computing resources to handle sophisticated computation tasks.

B. Mobility Model

Because of the constant mobility and speed variation, the location of vehicles and the distance between them changes continuously, which impacts the connection between task vehicles and service vehicles as well as the selection of vehicle that tasks are offloaded to [29]. Therefore, mobility is a critical factor that must be taken into account in this study. In this work, we assume that the vehicles are moving on the highway, which is a typical two-way road. The speed of vehicles changes continuously and satisfies the Poisson distribution [30].

For vehicles i and j, we can use $(x_i(\tau), y_i(\tau))$ and $(x_j(\tau), y_j(\tau))$ to denote their location (2D coordinates x and y) at time τ on a 2D map, respectively. Their inter-vehicle distance $d_{i,j}(\tau)$ can be calculated by (1):

$$d_{i,j}(\tau) = \sqrt{(x_i(\tau) - x_j(\tau))^2 + ((y_i(\tau) - y_j(\tau))^2} \quad (1)$$

Also, the distance $d_{i,r}(\tau)$ between vehicle i and edge server r can be calculated by (2):

$$d_{i,r}(\tau) = \sqrt{(x_i(\tau) - x_r(\tau))^2 + ((y_i(\tau) - y_r(\tau))^2} \quad (2)$$

C. Communication Model

Both V2V and V2I communications are used in the communication model. V2V stands for "vehicle-to-vehicle". It refers to the exchange of information between vehicles in close proximity to each other. On the other hand, V2I stands for "vehicle-to-infrastructure". It refers to the exchange of information between vehicles and roadside infrastructure. The vehicles without communication functions cannot build V2V and V2I communications because of their inability to communicate with edge servers and other vehicles, thus they cannot receive tasks from other vehicles and their tasks can only be executed locally.

In order to reduce energy consumption while ensuring communication range, in this work, the transmit power is adjusted over time based on the distances among computing entities. Transmit power refers to the amount of power used by a wireless device, such as a vehicle or an edge server, to transmit tasks over a wireless communication channel. It directly affects the power consumption of the wireless communication system, which can impact the battery life of the connected vehicles. A higher transmit power can extend the range of the wireless communication system and allow vehicles to communicate over a longer distance. However, it also consumes more power and drains the battery faster, reducing the driving range and overall performance of the vehicle. Therefore, adjusting transmit power is crucial for ensuring reliable and efficient communications among connected vehicles. The transmit power needs to be optimized based on various factors such as the communication distance and power consumption to achieve the best performance and efficiency.

For simplification, it is assumed that the transmission range is the same for every vehicle in our work. The radius of the coverage range is d.

1) V2V data transmission rate: Firstly, the signal-to-noise ratio (SNR) between vehicle i and vehicle j can be calculated by (3):

$$SNR_{i,j}(\tau) = \frac{p_{i,j}(\tau) \cdot G_{i,j}(\tau)}{\xi_{i,j}(\tau) \cdot d_{i,j}(\tau) \cdot \sigma_{i,j}(\tau)^2} \quad (3)$$

where $p_{i,j}(\tau)$ represents the transmit power when transmitting tasks from vehicle i to vehicle j, $G_{i,j}(\tau)$ represents the channel gain of antennas used for transmission, $\xi_{i,j}(\tau)$ is the path loss, $\sigma_{i,j}(\tau)^2$ represents the power of the white Gaussian noise, and $d_{i,j}(\tau)$ represents the distance calculated by (1).

Then, the instantaneous data transmission rate $R_{i,j}$ between vehicles i and j can be calculated by (4):

$$\begin{aligned} &R_{i,j}(\tau) = \eta_j \cdot B_i \cdot \log_2(1 + SNR_{i,j}(\tau)) \\ &s.t. \sum_{j=0}^{V} \eta_j \leq 1 \end{aligned} \quad (4)$$

where B_i is the total bandwidth of vehicle i, η_j represents the allocation proportion of the bandwidth to vehicle j, the sum of the allocated bandwidth cannot exceed the total bandwidth of vehicle i. In this work, the full available bandwidth is utilized for transferring tasks. By utilizing the full available bandwidth, more data can be transmitted simultaneously, resulting in higher data transmission rates.

2) V2E data transmission rate: Firstly, the interference noise $I_{i,r}(\tau)$ depends on other V2E connections and can be measured by (5):

$$I_{i,r}(\tau) = (N_r(\tau) - 1) \cdot p_{i,r}(\tau) \cdot g_r(\tau) \quad (5)$$

where $(N_r(\tau) - 1)$ represents the number of vehicles connecting with the edge server r (except the vehicle i itself),

$p_{i,r}(\tau)$ represents the allocated transmit power of the vehicle i, and $g_r(\tau)$ represents the channel gain.

Then, the data transmission rate between vehicle i and edge server r can be calculated by (6):

$$R_{i,r}(\tau) = \eta_i \cdot B_r \cdot \log_2 (1 + \frac{p_{i,r}(\tau) g_r}{I_{i,r}(\tau) + \sigma^2}) \tag{6}$$

where B_r is the available bandwidth of edge server r, η_i represents the allocation proportion of the bandwidth to vehicle i, and σ^2 represents the power of the white Gaussian noise.

3) E2C data transmission rate: The interference noise $I_{r,cs}(\tau)$ from other E2C connections can be measured by (7):

$$I_{r,cs}(\tau) = \sum_{s \neq r, s \in S} p_s(\tau) \cdot g_r(\tau) \tag{7}$$

where $p_s(\tau)$ represents the transmit power of edge server s. Thus, the data transmission rate between edge server r and cloud server cs can be calculated by (8):

$$R_{r,cs}(\tau) = B_{r,cs} \cdot \log_2 (1 + \frac{p_r(\tau) g_r(\tau)}{I_{r,cs}(\tau) + \sigma^2}) \tag{8}$$

where $B_{r,cs}$ represents the bandwidth between the cloud server cs and the edge server r.

D. Task Model

In this work, we assume that there are V vehicles and S edge servers. Set $\{1, 2, \cdots, V\}$ and $\{1, 2, \cdots, E\}$ represent the index set of vehicles and edge servers, respectively. Each vehicle generates a different number of data processing tasks, depending on the service request s. Some kinds of services require the collaboration of several sensors, then it can be divided into a small number of tasks. In contrast, for the service that requires many sensors, it can be divided into a big number of tasks. In this work, we assume that the number of tasks generated from one vehicle is K, which is not a fixed value and differs from different vehicles.

The task k on the vehicle v can be donated by a tuple $\langle C_{v,k}, S_{v,k}, P_{v,k} \rangle$, where $C_{v,k}$ represents the total number of CPU cycles required to complete the computing job, $S_{v,k}$ represents the amount of data to be processed, $P_{v,k}$ represents the priority assigned to the task k. Each task can be executed locally or offloaded to edge servers, idle vehicles, and the cloud server. The scheduling decision α can be denoted by $\alpha_{v,k} \in \{0, 1, 2, 3\}$, where $\alpha_{v,k}$ represents the scheduling decision for task k generated from vehicle v. $\alpha_{v,k} = 0$ and $\alpha_{v,k} = 1$ represent that the task is offloaded to an edge server and another vehicle, respectively. $\alpha_{v,k} = 2$ represents that the task is executed locally. $\alpha_{v,k} = 3$ represents that the task is transferred to the cloud server. The offloading decision for all the tasks on the vehicle v can be denoted by $\alpha = \{\alpha_{v,1}, \ldots, \alpha_{v,2}, \ldots, \alpha_{v,K}\}$.

Based on the scheduling decision, the tasks scheduled to the vehicle v can be denoted by set $L_v = \{l_1, l_2, \ldots, l_{ns}\}$, where ns represents the number of tasks allocated to the vehicle v, including the tasks generated by vehicle v and executed locally as well as the tasks offloaded from other vehicles. Similarly, let $G_s = \{g_1, g_2, \ldots, g_{ms}\}$ denote the tasks allocated to edge server s, where ms is the number of tasks. Let $H_{cs} = \{h_1, h_2, \ldots, h_{ls}\}$ denote the tasks allocated to the cloud server cs, where ls is the number of tasks. In addition, the tasks in the set are queued up in the order of priority, then the tasks can be executed based on the order. For example, in the set L_v, the priority and the corresponding execution order are denoted by $l_1 \geq l_2 \geq l_3 \geq \cdots \geq l_{ns}$.

E. Computation Model

For the task computation that happens after the task scheduling, both completion time and energy consumption are critical and need to be considered.

1) Local computing: The completion time $T^L_{v,k}$ and energy consumption $E^L_{v,k}$ of task k executed locally on the vehicle v can be defined as follows:

$$T^L_{v,k} = T_{v,l_{ns}} + \frac{C_{v,k}}{f_v} \tag{9}$$

$$E^L_{v,k} = \xi \cdot (f_v)^\gamma \cdot C_{v,k} \tag{10}$$

where $T_{v,l_{ns}}$ represents the completion time of the preceding task, l_{ns} represents the index of the preceding task, ns represents the number of existing tasks in the processing queue at vehicle v, f_v is the computing power of vehicle v, representing the number of CPU cycles that one vehicle can execute per second, and ξ and γ are the vehicle power consumption coefficients.

2) Edge Computing: Apart from local execution, the tasks can also be offloaded to either edge servers or other idle vehicles. For the offloaded task k, the execution time $T^O_{v,k}$ is comprised of task transmission time $T^{O,trans}_{v,k}$, task processing time $T^{O,exe}_{v,k}$ and result feedback time $T^{O,fb}_{v,k}$:

$$T^O_{v,k} = T^{O,trans}_{v,k} + T^{O,exe}_{v,k} + T^{O,fb}_{v,k} \tag{11}$$

In (11), the task transmission time $T^{O,trans}_{v,k}$, from vehicle v to either vehicle j or edge server r, can be calculated by:

$$T^{O,trans}_{v,k} = \begin{cases} T^{V2V,trans}_{j,l_{ns}} + \frac{S_{v,k}}{R_{v,j}}, & \alpha_{v,k} = 1 \\ T^{V2E,trans}_{r,g_{ms}} + \frac{S_{v,k}}{R_{v,s}}, & \alpha_{v,k} = 0 \end{cases} \tag{12}$$

where $T^{V2V,trans}_{j,l_{ns}}$ and $T^{V2E,trans}_{s,g_{ms}}$ represent the transmission completion time of the preceding task at vehicle j and edge server s, respectively. l_{ns} and g_{ms} represent the index of the preceding task at vehicle j and edge server s, respectively, ns and ms represent the number of existing tasks in the transmission queue at vehicle j and edge server s, respectively.

The task processing time $T^{V2E,exe}_{v,k}$ can be calculated by:

$$T^{O,exe}_{v,k} = \begin{cases} T^{V2V,exe}_{j,l_{ns}} + \frac{C_{v,k}}{f_j}, & \alpha_{v,k} = 1 \\ T^{V2E,exe}_{s,g_{ms}} + \frac{C_{v,k}}{f_s}, & \alpha_{v,k} = 0 \end{cases} \tag{13}$$

where $T^{V2V,exe}_{j,l_{ns}}$ and $T^{V2E,exe}_{s,g_{ms}}$ represent the completion time of the preceding task at vehicle j and edge server s,

respectively, l_{ns} and g_{ms} represent the index of the preceding task, ns and ms represent the number of existing tasks in the processing queue.

In this work, we assume that the movement of vehicles is ignored during the task offloading and result feedback. In addition, the result feedback time $T_{v,k}^{V2E,fb}$ can be ignored because of the small data size of computed results [13].

In addition to the completion time, the energy consumption of the task computation can be calculated by:

$$E_{v,k}^{O} = \begin{cases} p_v \cdot \frac{S_{v,k}}{R_{v,j}} + \xi \cdot (f_j)^{\gamma} \cdot C_{v,k}, & \alpha_{v,k} = 1 \\ p_v \cdot \frac{S_{v,k}}{R_{v,s}}, & \alpha_{v,k} = 0 \end{cases} \tag{14}$$

where p_v is the transmit power of vehicle v, representing the amount of data that can be transmitted per second.

3) Cloud Computing: In addition to local computing and edge computing, the time-tolerant and computation-intensive tasks can be offloaded to the cloud server. The total execution time of task k is expressed by:

$$\begin{aligned} T_{v,k}^{CS} = T_{v,k}^{V2E,trans} + T_{v,k}^{E2C,trans} + T_{v,k}^{CS,exe} \\ + T_{v,k}^{E2C,fb} + T_{v,k}^{V2E,fb} \end{aligned} \tag{15}$$

where $T_{v,k}^{V2E,trans}$ and $T_{v,k}^{E2C,trans}$ represent the transmission time of the task k from vehicle v to the edge server and from the edge server to the cloud server, respectively, $T_{v,k}^{CS,exe}$ represents the task processing time on the cloud server, $T_{v,k}^{E2C,fb}$ and $T_{v,k}^{V2E,fb}$ represent the feedback time of the computed results from the cloud server to the edge server and from the edge server to the vehicle v, respectively. Similar to the edge computing, the result feedback time $T_{v,k}^{E2C,fb}$ and $T_{v,k}^{V2E,fb}$ can be ignored.

In (15), the task transmission time $T_{v,k}^{E2C,trans}$, from the edge server r to the cloud server cs, can be calculated by:

$$T_{v,k}^{E2C,trans} = T_{cs,h_{ls}}^{E2C,trans} + \frac{S_{v,k}}{R_{r,cs}} \tag{16}$$

where $T_{cs,h_{ls}}^{E2C,trans}$ represents the transmission completion time of the preceding task at cloud server cs, h_{ls} represents the index of the preceding task at cs, and ls represents the number of existing tasks in the transmission queue at cs.

The task processing time can be calculated by:

$$T_{v,k}^{E2C,exe} = T_{cs,h_{ls}}^{E2C,exe} + \frac{C_{v,k}}{f_s} \tag{17}$$

where $T_{cs,h_{ls}}^{E2C,exe}$ represents the completion time of the preceding task at the cloud server cs, h_{ls} represents the index of the preceding task, ls represents the number of existing tasks in the processing queue.

F. Problem Formulation

In this work, minimizing the overall completion time T and energy consumption E jointly is the primary target of the proposed collaborative multi-task scheduling system. In order to achieve the target, the objective function is defined as a weighted sum of the completion time and energy consumption. The completion time is defined as the overall time taken to fulfil all the tasks in one service request, determined by the last computing entity to finish. Energy consumption, on the other hand, is defined as the sum of energy consumption on all vehicles. The objective function is formulated as (18):

$$\begin{aligned} obj: & min\left\{\omega_1 \cdot T + \omega_2 \cdot E\right\} \\ & T = max\left\{T_{L_1}, \ldots, T_{L_V}, \ldots, T_{E_1}, \ldots, T_{E_S}, \ldots, T_{CS}\right\} \\ & E = \sum_{v=1}^{V} \sum_{k=1}^{K} E_{v,k} \\ s.t. \quad & C_1: \alpha_{v,k} \in \{0,1,2,3\} \\ & C_2: d_{v,j}^{O} < d_v, \quad \forall v, j \in V \\ & C_3: d_{v,s}^{O} < d_s, \quad \forall v \in V, \forall s \in S \\ & C_4: T_{v,k} \leq D_{v,k} \\ & C_5: E_{v,k} \leq E_{max} \end{aligned} \tag{18}$$

where ω_1 and ω_2 are weight factors, T_{L_V} and T_{E_S} represent the completion time of all the tasks in vehicle V and edge server S, respectively. T_{CS} represents the completion time of the tasks that are allocated to the cloud server. $E_{v,k}$ denotes the energy consumption to process the task k generated from vehicle v. The objective function is subjected to the following constraints:

C_1 implies that each task k can be executed locally, offloaded to an edge server or another vehicle, or offloaded to the cloud server. C_2 ensures that the vehicle v can offload tasks to another vehicle j that in its communication range d_v. C_3 ensures that the vehicle v can offload tasks to the edge server r that in the coverage range d_s of edge server s. C_4 ensures that the task k must be completed in the required time $D_{v,k}$. C_5 ensures that the energy consumption for task computation should be less than the maximum energy budget E_{max}.

IV. System Design

In this section, the proposed MEPPO algorithm is introduced, incorporating the transmit power allocation strategy, offloading target selection strategy, and PPO-based dynamic task offloading and priority assignment strategy.

A. Task Scheduling Procedure

The task scheduling procedure is illustrated in Fig. 2. Firstly, as the number of vehicles within the communication range of the edge server changes dynamically, the edge server constantly monitors the presence and connectivity status of vehicles. This ensures that it has an up-to-date view of the vehicles that can potentially offload tasks to the edge server. Secondly, the vehicles continuously generate service requests, with varying numbers of tasks associated with different requests. The edge server collects information about the tasks, including their computational requirements and data sizes $\langle C, S \rangle$. This information allows the edge server to evaluate the feasibility and potential benefits of offloading

tasks from the vehicles to itself or to other computing entities within the network. Next, based on the task information and the state of the computation entities, including their computational capabilities and current workload, the edge server makes task scheduling decisions, including task offloading and priority assignment. Task offloading determines which tasks should be offloaded and which tasks should be executed locally on the vehicles. Furthermore, the edge server assigns execution priority to the tasks that are offloaded to itself or other vehicles. The goal is to optimize resource utilization, minimize latency, and balance the computational load across the computing entities. Finally, the computation results are fed back and fused to generate the final vehicle control decision.

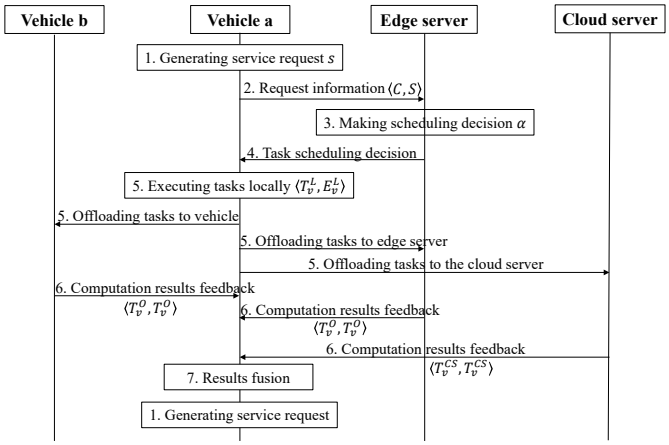


Fig. 2: The flow chart of task scheduling framework

B. Transmit Power Allocation Strategy

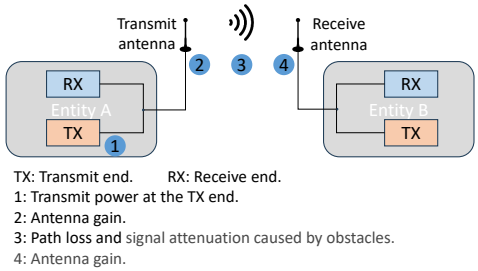


Fig. 3: Signal transmission

In this work, the transmit power is allocated according to the distance between the vehicle and the task receiver. The illustration of signal transmission is shown in Fig. 3. The data is converted into radio signals and then the radio signals are transmitted from the transmit (TX) end to the receive (RX) end [31]. The strength of signals gradually attenuates during wireless transmission. The strength of signals received by the entity B from A is calculated by $signal\ strength = 1+2-3+4$. Therefore, received signal strength indicator $RSSI$ can be calculated by:

$$RSSI = P + TX - (L + SA) + RX \tag{19}$$

where L denotes the path loss in dB, P denotes the transmit power, $RSSI$ denotes the received signal strength indicator, SA denotes the signal attenuation caused by obstacles, TX and RX signify the transmit end's antenna gain and receive end's antenna gain, respectively. The relationship between path loss and signal transmission distance can be represented by:

$$L = 32.4 + 26 \lg (d) + 20 \lg (f) \tag{20}$$

where d denotes the transmission distance (m) and f represents working frequency (MHz).

Then, according to (19), (20), and the maximum transmit power P_{max} limited by hardware devices, the maximum transmission distance d_{max} can be determined by $d_{max} = 10^{\frac{P_{max}-RSSI+TX-SA+RX-32.4-20\lg(f)}{26}}$.

At last, the allocated transmit power can be changed within the maximum range P_{max} according to the distance to the task receiver, denoted by $p = d \cdot \frac{P_{max}}{d_{max}}$.

C. Vehicle Selection Strategy

In this work, the marking scheme is designed to select the vehicle with the highest score as an optional offloading target. Firstly, the information of the vehicles around the task vehicle is collected, including the distances to the task vehicle and the workload. The set of collected information can be denoted by:

$$\begin{Bmatrix} d_1 & d_2 & \dots & d_n \\ w_1 & w_2 & \dots & w_n \end{Bmatrix} \tag{21}$$

where n represents the number of vehicles in the communication range of the task vehicle, d_i and w_i represent the distance and the workload of vehicle i, respectively.

Then, the score of each vehicle can be determined by:

$$s_i = \omega_1 \cdot \frac{d_i - d_{min}}{d_{max} - d_{min}} + \omega_2 \cdot \frac{w_i - w_{min}}{w_{max} - w_{min}} \tag{22}$$

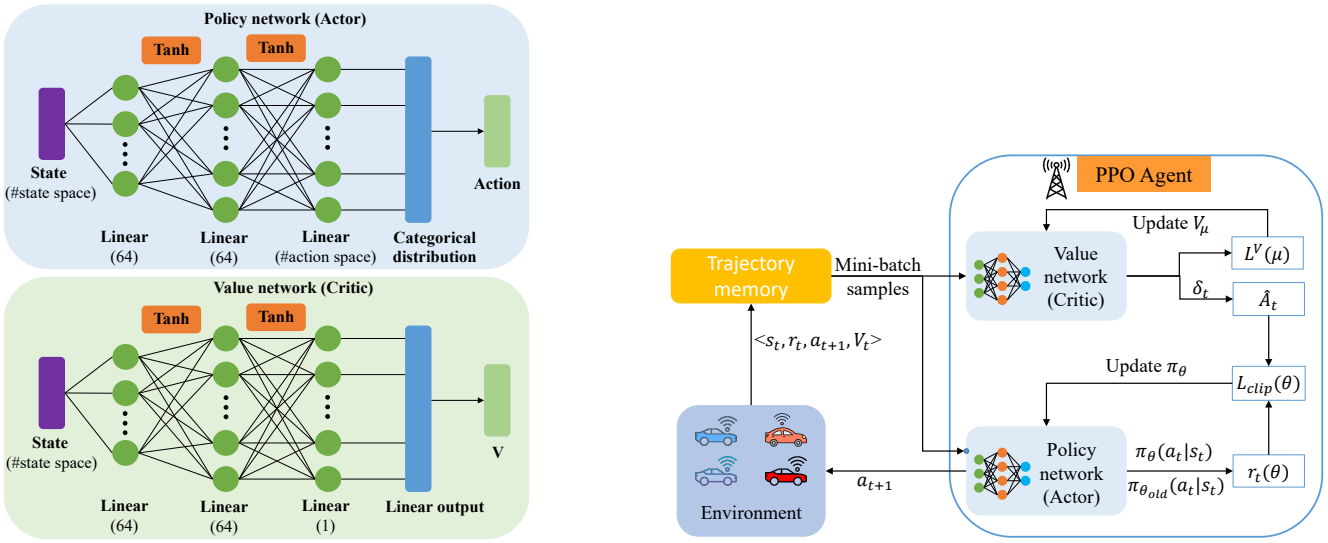
where s_i is the score of vehicle i, ω_1 and ω_2 represent the weight values. As shown in (22), the score is calculated according to the weighted sum of normalized distance and workload.

D. Reinforcement Learning for Task Scheduling

With the rapid development of deep learning, training Large Models for multiple tasks becomes a popular paradigm. The large models have both better performance on existing tasks and the ability to develop new kinds of skills. Motivated by this paradigm and in order to ensure adaptability to dynamic environments, this work focuses on training a model for vehicles with a large number of tasks. This approach allows the trained big model to be adapted to a wide range of vehicles with various tasks.

1) Training environment: In this work, the edge server acts as the agent in the reinforcement learning framework for making task scheduling decisions. Using reinforcement learning, the edge server learns from its interactions with the environment, which includes the dynamic number of vehicles, their service requests, and the available computational resources.

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2023.3321679



(a) Architecture of the Policy and Value networks.

(b) The structure of the Proximal Policy Optimization algorithm

Fig. 4: Network architecture

2) States: The states represent the current situation of the environment that the agents are in. In this work, the states are represented as a set of observations that the vehicles receive from the environment, including the positions $P^e(t)$ of the vehicles and edge servers, the information $I^v(t)$ of tasks generated from each vehicle, the communication overhead $M^v(t)$ of each vehicle, and the computation workload $C^e(t)$ of each computing entities. Therefore, the state space of each agent can be represented by:

$$s_t = \{P^e(t), I^v(t), M^v(t), C^e(t)\} \tag{23}$$

3) Actions: The actions are the decisions that the agents make in response to the current state of the environment. Different from other studies, in this work, the combination action is represented as a set of multiple actions that control the task scheduling behaviour of each vehicle, including the offloading decisions and the priority assignment decisions:

$$\begin{aligned} &A_t^v = \left\{a_1^1, \ldots, a_1^K, \ldots, a_i^k, \ldots, a_V^K\right\}, \\ &a_i^k = \left\{\alpha_i^k, p_i^k\right\} \\ &\forall i \in \{1, \ldots, V\}, \\ &\forall k \in \{1, \ldots, K\}, \alpha_i^k \in \{0, 1, 2, 3\} \end{aligned} \tag{24}$$

where a_i^k denotes the combination action for task k generated from vehicle i, including the offloading decision α_i^k and the assigned execution priority p_i^k. Thus, a two-dimensional coordinate system is used to describe the action space. The x-axis represents the offloading of tasks, the y-axis represents the assignment of priority.

4) Reward: The rewards represent the feedback that the agents receive from the environment based on the actions they take. The rewards are typically designed to encourage the agents to take actions that lead to desirable outcomes, such as minimizing request completion time and energy consumption.

In this work, the reward function is defined as:

$$r = \alpha \times \frac{exp_{time}}{max_{exp_{time}}} + \beta \times \frac{exp_{energy}}{max_{exp_{energy}}} \tag{25}$$

where exp_{time} represents the expected completion time of the service request, exp_{energy} represents the expected energy consumption of that request, $max_{exp_{time}}$ and $max_{exp_{energy}}$ represent the maximum exp_{time} and exp_{energy} the agent had reached. α and β are constants controlling the weight of the corresponding variable.

5) Dynamic space: In order to achieve adaptability to the dynamic environment, this work involves training the scheduling method with a large number of tasks. Once the model is trained, the state space is supplemented with a null task if the actual number of tasks is fewer than the required number of tasks when the model is deployed. This approach ensures that the scheduling method remains environment-adaptive even when faced with varying task numbers in the dynamic scenario. The task supplement scheme is defined as:

$$\begin{aligned} S &= \begin{cases} I^v_{n\times 2}, & \text{if } n = K \\ I^v_{K\times 2}, & \text{if } n < K \end{cases} \\ I^v_{n\times 2} &= \begin{bmatrix} C_{v,1} & C_{v,2} & \ldots & C_{v,n} \\ S_{v,1} & S_{v,2} & \ldots & S_{v,n} \end{bmatrix}^T \\ I^v_{K\times 2} &= \begin{bmatrix} C_{v,1} & \ldots & C_{v,n} & C_{v,n+1}=0 & \ldots & C_{v,K}=0 \\ S_{v,1} & \ldots & S_{v,n} & S_{v,n+1}=0 & \ldots & S_{v,K}=0 \end{bmatrix}^T \end{aligned} \tag{26}$$

where K represents the required number of tasks, n is the actual number of tasks in one service request, and S is part of the state space, representing task information I^v.

E. Input and output

As shown in Fig. 4a, the policy network is a Multi-Layer Perceptron (MLP)-styled neural network consisting of two hidden layers and an output layer. Two hidden layers are with 64 neurons (Linear mapping) and the Tanh activation function throughout. Instead of using the softmax function, the output layer samples a discrete value from the categorical distribution constructed by given logits (the output from the hidden layers), which is then transformed into an action. In summary, the policy network is used to predict action based on the provided state. Therefore, the input of the policy network is the observation of the environment, and the output is the assignment of tasks. For example, as shown in Fig. 5, when making task scheduling decisions for a vehicle with 10 tasks, the size of the input (state) is 19, including the positions p, communication workload m, and computation workload c of the vehicle v, edge server e and cloud server s as well as the information of 10 tasks (based on Eq. 23). The size of the output (action) is 20, including the offloading decision α and execution priority p for all 10 tasks (based on Eq. 24).

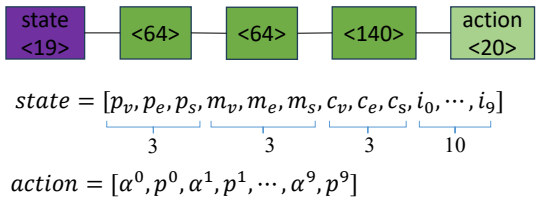


Fig. 5: Illustration of input and output

F. Proximal Policy Optimization Algorithm

The structure of the PPO algorithm is shown in Fig. 4b. As shown in the figure, the actor and critic are two components that work together to optimize the policy and value function in reinforcement learning. The actor is responsible for learning and updating the policy π_θ, which determines the agent's actions based on the observed states. The critic is responsible for learning and estimating the value function V_μ, which evaluates the quality or expected cumulative reward of being in a particular state. The core idea behind PPO is to introduce a "proximal" term to the objective function, which constrains the policy update to a certain proximity to the old policy. By maintaining this proximity, PPO ensures that the policy changes are not too drastic, which helps to maintain stability during the learning process. The PPO algorithm consists of the following five steps:

1) Collecting trajectories: Initially, a set of trajectories is collected by executing the current policy in the environment. These trajectories consist of states, actions, rewards, and other relevant information.

2) Computing surrogate objective: Because PPO is based on Trust Region Policy Optimisation (TRPO) [9], the update is monotonic, where the updated policy is always better than the previous one. The objective function in TRPO is denoted by:

$$L_{cpi}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t \right) \right] \tag{27}$$

where cpi represents the conservative policy iteration, the probability ratio $r_t(\theta)$ can be denoted by:

$$r_t(\theta) = \frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \tag{28}$$

which represents the importance sampling weight between the old policy $\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)$ and new policy $\pi_\theta \left(a_t \mid s_t \right)$.

However, maximization of L_{cpi} may introduce a large variance and lead to an excessively large policy update. Therefore, the surrogate objective function used in this work is a modification of the policy's objective function, designed to ensure stable policy updates, which is denoted by:

$$L_{clip}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right] \tag{29}$$

As shown in (29), the objective function is defined as the minimum between the unclipped surrogate objective and the clipped surrogate objective, comparing the probability ratio with a clipped version of the ratio. The clipped surrogate objective can be denoted by

$$\text{clip} \left(r_t(\theta), 1-\epsilon, 1+\epsilon \right) = \begin{cases} 1-\epsilon, & \text{if } r_t(\theta) \leq 1-\epsilon \\ 1+\epsilon, & \text{if } r_t(\theta) \geq 1+\epsilon \\ r_t(\theta), & \text{otherwise} \end{cases} \tag{30}$$

which bounds the policy update to a certain range $[1-\epsilon, 1+\epsilon]$, preventing it from deviating too far from the old policy.

3) Estimating policy gradient: The policy gradient is estimated by computing the gradient of the surrogate objective with respect to the policy parameters. This is typically done using automatic differentiation.

4) Performing optimization: The estimated policy gradient is used to update the policy parameters. PPO typically employs stochastic gradient descent (SGD) to perform the parameter updates. Multiple iterations of this step can be performed to improve the policy over time. Specifically, the policy is run for T timesteps and then the collected samples are used for an update. This work requires an advantage estimator $\hat{A}$. The estimator used in this work is a truncated version of Generalized Advantage Estimation (GAE):

$$\hat{A}_t = \sum_{k=0}^{T-1} (\gamma\lambda)^k \delta_{t+k} \tag{31}$$

where $\hat{A}_t$ represents an estimator of the advantage function at timestep t, $\delta_t = r_t + \gamma V_{\theta_v}(s_{t+1}) - V_{\theta_v}(s_t)$, γ and λ are the discount factor of future rewards to control the variance of the advantage function.

5) Repeating the process: The above steps are repeated iteratively, with new trajectories collected, the surrogate objective computed, the policy gradient estimated, and the policy parameters updated. This iterative process allows the

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2023.3321679

policy to improve incrementally while maintaining stability through the use of the proximal term.

The training process of the PPO algorithm is shown in Algorithm 1.

Algorithm 1 Training process of the PPO algorithm used for generating task offloading and priority assignment decisions

```
Initialize policy parameters θ_0
for iteration=1,2,... do
  Collect an episode to replay memory D.
  for actor=1 to N do
    for timestep t = 1 to T do
      Observe the state s_t of the environment.
      Select an action a_t based on the observed state.
      The environment transitions to a new state s_{t+1}
      based on the selected action.
      Receive a reward r_{t+1} based on the new state.
      Compute advantage estimate Â_t using Eq. (31).
    end for
  end for
  Cache all sampled data in the replay set.
  Update L_clip(θ) using the sampled data.
  π_{θ_old} ← π_θ
end for
```

6) Illustration of the MEPPO algorithm: One example is provided to illustrate the task scheduling and execution based on the designed MEPPO algorithm. As shown in Fig. 6, there are four vehicles in the range of the edge server, including three task vehicles (TaV) and one service vehicle (SeV). All task vehicles generate varying numbers of tasks. Firstly, one of the vehicles moving around the task vehicle is selected as one of the optional task offloading targets based on the vehicle selection strategy. (As shown in Fig. 6, SeV D is selected as an offloading target for TaV A and B.) Secondly, the MEPPO-based scheduling method generates task offloading decisions and priority assignment decisions for each task vehicle. Then, the tasks on the vehicle are offloaded according to the task offloading decisions and each task is assigned an execution priority based on the priority assignment decisions. (As shown in Fig. 6, $A_{1,2}$ represents that the task 1 of vehicle A is offloaded to the edge server with priority 2.) Thirdly, among the task offloading, the tasks are offloaded with transmit power allocated by transmit power allocation strategy. Fourth, after task offloading, the tasks on the same computing entity are executed in order, from high priority to low priority. (As shown in Fig. 6, Four tasks are offloaded to the edge server and executed from priority 4 to 1.) The workflow of the MEPPO-based energy-efficient and priority-aware task scheduling method is shown in Algorithm 2.

V. Experiment

In this section, the performance of the proposed method is evaluated via simulation.

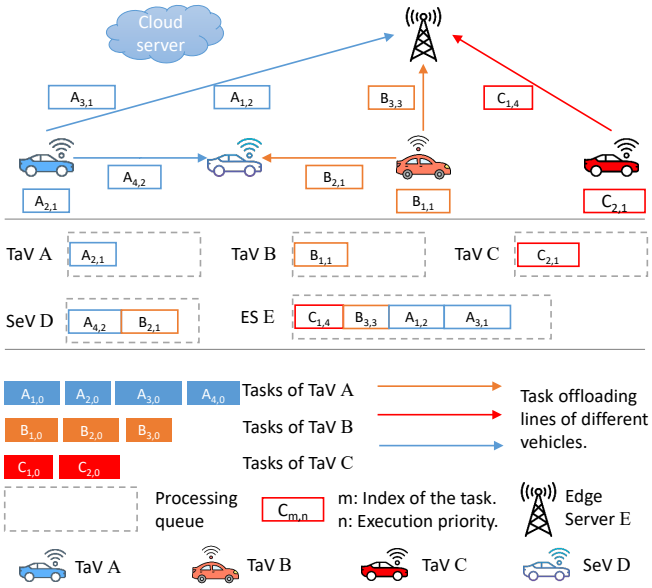


Fig. 6: One illustration of task scheduling and execution

Algorithm 2 Workflow of the MEPPO-based energy-efficient and priority-aware task scheduling method

```
The trained model is deployed on the edge server.
for timestep t = 1,2,... do
  Monitor the number of vehicles V around the edge server.
  for vehicle v = 1 to V do
    Collect the state information s_t of vehicle v according to (23).
    Determine the SeV for vehicle v based on the Vehicle Selection Strategy.
    Collect the number of tasks K in the service request generated from vehicle v.
    for task k = 1 to K do
      Generate task offloading decision and determine execution priority for each task k based on the PPO algorithm.
      if k is offloaded then
        Allocate transmit power to k based on the Transmit Power Allocation Strategy.
      end if
    end for
  end for
end for
```

A. Experimental Setting

1) The map: In the simulation, we assumed that all the vehicles are moving on the two-way highway, and the eight edge servers are deployed alongside the road. All the vehicles are keeping moving forth along the lanes. The two-way highway is a typical scenario to evaluate the task scheduling algorithm [32]. The distribution of vehicles is shown in Fig. 7, in which the size of node area corresponds to the number of tasks in the service request generated

from the vehicle and the colour of the node represents the computation resource required to complete the request.

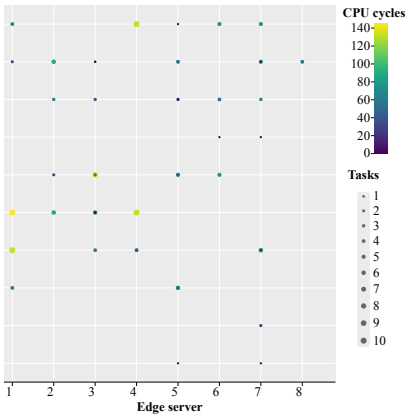


Fig. 7: The distribution of vehicles

2) Vehicles and tasks: The task scheduling ability of the edge server is evaluated under five testing scenarios with different numbers of vehicles, from 10 vehicles to 50 vehicles. Each vehicle can generate service requests continuously, with a different number of tasks in the request, from 0 to 10. The distribution of various service requests with different numbers of tasks is shown in Fig. 8. As can be seen from the figure, each scenario comprises some service vehicles that do not generate tasks (0 tasks), the maximal number of tasks in one request is 10 and most requests comprise 2-6 tasks.

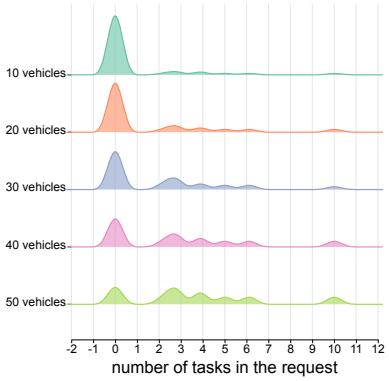


Fig. 8: The distribution of tasks in service requests

3) Parameters: In the experiments, the deep reinforcement learning algorithms are trained on a server with Intel Xeon W-22555 (10 cores, 3.70GHz), and NVIDIA RTX 3070 GPU. The simulations are conducted on a PC of Windows 10 with Intel i5-1035G1 and 16 GB DRAM. The detailed experimental parameters are listed in Table III. Among them, the Receive antenna gain RX is determined according to the study about the antenna performance analysis for V2X communication of connected vehicles [33].

TABLE III: Experimental parameters

Parameters	Value
Number of vehicles	10-50
Number of edge servers	8
Number of tasks	0-10
Required CPU cycles of the tasks C	2-20 cpu cycles
Computation power of edge servers	2 cpu cycles/s
Computation power of vehicles	1 cpu cycles/s
data size S	2-20 Mb
Bandwidth of edge server	100 MHz
Speed of vehicles	$\approx$ 25 m/s
Transmit power of vehicles	1 dBm
Execution power of vehicles	3-4 dBm
White Gaussian noise	-174 dBm/Hz
Power consumption coefficient ξ	10^{-11}
Power consumption coefficient γ	2
Received Signal Strength Indicator $RSSI$	-65 dBm
Transmit antenna gain TX	20 dBi
Receive antenna gain RX	-8 dBi
Signal attenuation SA	7 dB
Working frequency f	5GHz
Learning rate	0.0003
Size of Mini-batch	32
Number of steps in each episode	2048
Entropy loss coefficient	0.01

B. Comparative Methods

In the experiments, the details of the compared methods are as follows:

a) Deep Deterministic Policy Gradient (DDPG) [20]: The DDPG-based scheduling method mainly aims to optimize the request completion time.

b) Soft Actor-Critic (SAC) [21]: In order to achieve a joint optimization of completion time and energy usage, the SAC is utilized to design an efficient task scheduling scheme.

c) Random scheduling: The random-based scheduling method randomly allocates all the data processing tasks in the service request to edge servers or vehicles.

d) Offloading-only scheduling: With the offloading-only scheduling method, all the tasks in one request are assigned to edge servers and the cloud server.

e) Local-only scheduling: In contrast to the offloading-only method, with the local-only scheduling method, all the tasks are executed locally on the Task Vehicle itself.

C. Performance Evaluation

1) Convergence: During the training of reinforcement learning, the agent aims to maximize its cumulative reward over time. By exploring the environment, trying different actions, and observing the resulting rewards, the agent learns to take actions that lead to higher rewards and avoid actions that lead to lower rewards. This learning process helps the agent to converge to an optimal policy that maximizes its long-term cumulative reward. As shown in Fig. 9, the proposed MEPPO-based method can achieve a fast convergence under different numbers of vehicles after around 500 episodes.

2) Request completion time: Request completion time represents the overall time taken to fulfil all the tasks in one service request, depending on the last completed task.

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2023.3321679

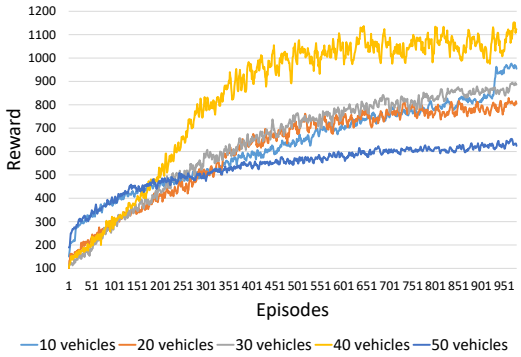


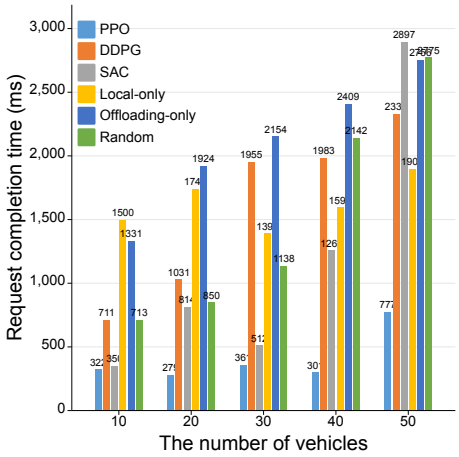
Fig. 9: Comparison of the reward

Fig. 10a illustrates the comparison of total completion times for ten requests among six different methods. As shown in the figure, the utilization of the MEPPO-based method for task scheduling demonstrates a noteworthy reduction in latency compared to other scheduling methods. Conversely, the offloading-only scheduling method results in the longest request completion time among the alternatives. This is attributed to the offloading of all tasks to the edge server, resulting in an extended transmitting and waiting period before task execution. In addition, the increase in the number of vehicles does not always result in an increase in request completion time because the rise in the number of vehicles offers additional computational resources, facilitating rapid task processing.

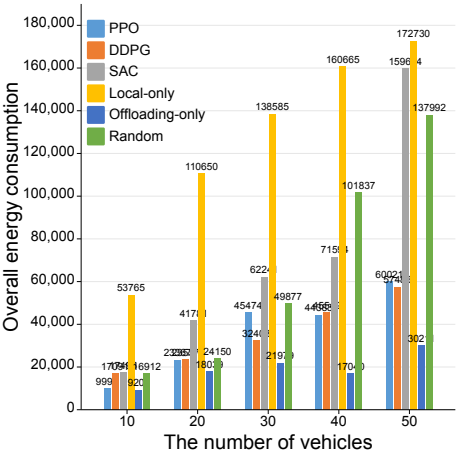
3) Overall energy consumption: Overall energy consumption represents the total energy consumption required for all the computing entities to complete the requests. The comparison of energy consumption among the six methods is shown in Fig. 10b. As observed in the figure, the local-only method incurs significantly higher energy for performing computation tasks compared to the other methods, because all the tasks are executed locally. In contrast, the offloading-only method exhibits the lowest energy consumption as executing tasks on the edge server consumes less energy than performing them locally on vehicles. Moreover, the MEPPO-based and DDPG-based methods result in relatively lower energy consumption, but the DDPG-based method requires more time than MEPPO to complete the requests.

4) Priority assignment: The comparison of request completion time between prioritized task execution and non-prioritized execution is depicted in Fig. 11. The results of the comparison indicate that assigning execution priority to tasks and performing them in a sequential order can significantly reduce the request completion time. Furthermore, the benefits of considering priority become more pronounced as the number of vehicles increases.

5) Transmit power allocation: Fig. 12 compares the consumption of transmission energy between transmitting tasks using dynamic and fixed transmit power. The results of the comparison demonstrate that adjusting the transmit power dynamically based on the distance can lead to a significant reduction in energy consumption. Moreover, as



(a) Request completion time



(b) Overall energy consumption

Fig. 10: Comparison on request completion time and energy consumption

the number of vehicles increases, the consumption of transmission energy initially decreases and then sharply increases. This trend can be attributed to the fact that as the number of vehicles increases, the distances between them become shorter, allowing for the use of low transmit power during task transmission. However, with a further increase in the number of vehicles, more tasks need to be offloaded to edge servers or distant vehicles, requiring higher transmit power and resulting in an overall increase in energy consumption.

6) Dynamic number of vehicles: In this section, the effectiveness of the proposed MEPPO-based scheduling method in terms of time saving and energy saving is assessed under the scenario that the traffic volume in one area changes continuously. The performances on time saving and energy saving can be quantified by:

$$TS = 1 - \frac{T_{meppo}}{T_{local}}, \ ES = 1 - \frac{E_{meppo}}{E_{local}} \tag{32}$$

where TS and ES represent the ratio of time saving and energy saving, respectively. T_{cost} and T_{under} represent the request completion time under the MEPPO-based task

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2023.3321679

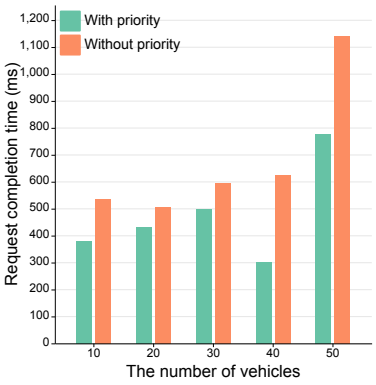


Fig. 11: The influence of priority assignment on request completion time

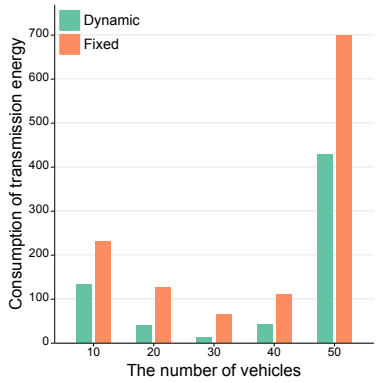


Fig. 12: The influence of dynamic transmit power allocation on energy consumption

scheduling and the local-only scheduling, respectively. Similarly, E_{cost} and E_{under} represent the energy consumption under the MEPPO-based task scheduling and the local-only scheduling, respectively.

In order to conduct the experiment, we simulated the traffic volume using real traffic data obtained from $NYC\ OpenData$ [34]. The data is collected by the New York City Department of Transportation (NYC DOT) using Automated Traffic Recorders (ATR). Fig. 13a illustrates the daily fluctuations in traffic volume at STATE STREET, Manhattan, recorded between Oct. 19, 2022 and Oct. 25, 2022. Then, based on this actual traffic volume data, the performance of the proposed method is evaluated and the result is depicted in Fig. 13b. The traffic trending in the figure represents the fluctuation of traffic volume on Oct. 19, which is normalized to [0, 1]. From the figure, it is evident that the vehicles follow a consistent pattern in terms of time and energy savings as the traffic volume fluctuates. Interestingly, this trend opposes the changes observed in traffic volume. Specifically, during periods of low traffic volume, particularly before 9 AM, vehicles on the road can complete service requests with shorter time and lower energy consumption compared to executing requests locally, with the ratio of time saving and energy saving higher than 0.6 and 0.3, respectively. However, as the traffic volume increases, between 10 AM and 7 PM, the time and energy costs rise significantly, resulting in a reduction in the ratio of time and energy savings.

In this section, the effectiveness of the proposed method in terms of time saving and energy saving is assessed under the scenario that the tasks generated by the vehicles change continuously. As shown in Fig. 13c, the proposed method demonstrates robust performance for the majority of the time, saving much time and energy compared to local task execution. Nonetheless, its performance tends to degrade when confronted with an exceptionally high volume of tasks because coping with a substantial task load requires more time and energy.

VI. Real-world Test

In addition to the simulation results presented in Section V, the on-site test is executed to evaluate the performance of the proposed scheduling method in this section.

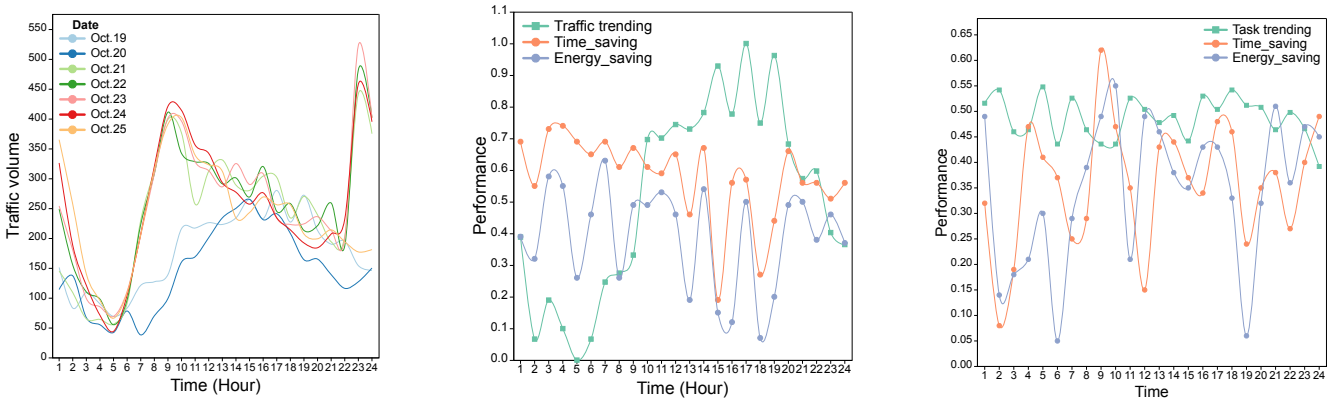
A. Experimental Setting

The on-site experiment was performed in a corridor, which was simulated to resemble a highway. Fig. 14a illustrates the scenario, where three robotic vehicles are moving in the same direction but at varying speeds. The laptop was placed along the road, serving as the simulated edge server. In this experiment, three robotic vehicles equipped with $Nvidia$ Jetson nano module can perform data processing tasks efficiently. The jetson nano module can provide sufficient computing capacity for in-vehicle services and applications. Each vehicle can generate requests constantly, consisting of four computation tasks. These tasks can be executed locally on the vehicle or offloaded to other computing entities via V2V and V2I communications. The communication architecture can be observed in Fig. 14b. According to the actual traffic scenario, four computation tasks are chosen: pedestrian detection, vehicle detection, traffic light detection, and lane detection.

B. Performance Evaluation

The performance of the proposed MEPPO-based scheduling method was evaluated by comparing the request completion time for each vehicle with that under the local-only scheduling method. The comparison results are illustrated in Fig. 15.

As depicted in Fig. 15, the request completion time for each vehicle is significantly reduced when using the MEPPO-based scheduling method compared to the local-only scheduling method. This demonstrates the effectiveness of the MEPPO-based method in reducing latency. Furthermore, under the local-only scheduling method, there is a noticeable discrepancy in the request completion time among the three vehicles due to variations in the actual task execution times. Conversely, the MEPPO-based scheduling method ensures that the requests are completed almost simultaneously, indicating its ability to achieve load balancing across all vehicles.



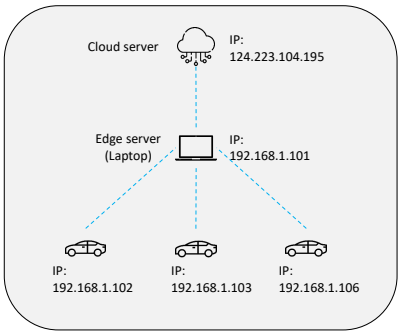
(a) The monitored traffic volume in one week.

(b) Time and energy saving under changing traffic volume.

(c) Time and energy saving under changing number of tasks.

Fig. 13: Comparison of the performances on time and energy saving under changing traffic volume.

(a) On-site test



(b) Architecture

Fig. 14: The scenario about on-site test

VII. Conclusion and Future Work

In this paper, we explored the challenges and opportunities in optimizing the completion time and energy consumption of multi-sensor in-vehicle services. Subsequently, an innovative joint task scheduling and resource allocation method is designed based on the proposed Multi-action and Environment-adaptive Proximal Policy Optimization algorithm. The proposed method surpasses existing studies by incorporating task prioritization and dynamic transmit power allocation to further reduce completion time and energy consumption. Additionally, the proposed method exhibits adaptability to the dynamic vehicular edge computing paradigm, enabling decision-making for varying numbers of vehicles. Furthermore, the method is evaluated via both simulation and on-site tests.

Currently, the primary focus of this work is on generating offloading decisions and allocating priority to independent tasks. However, some services can be divided into multiple tasks with task dependency. Therefore, refining the scheduling method to efficiently offload and prioritize dependent tasks is another challenge. In future work, we will continue to design task scheduling method to solve the problem of dependency-aware task offloading in vehicular edge computing.

References

[1] P. Li, X. Wang, K. Huang, Y. Huang, S. Li, and M. Iqbal, "Multi-model running latency optimization in an edge computing paradigm," *Sensors*, vol. 22, no. 16, p. 6097, 2022.

[2] F. M. Ortiz, M. Sammarco, L. H. M. Costa, and M. Detyniecki, "Applications and services using vehicular exteroceptive sensors: a survey," *IEEE Transactions on Intelligent Vehicles*, 2022.

[3] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang *et al.*, "Planning-oriented autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 853–17 862.

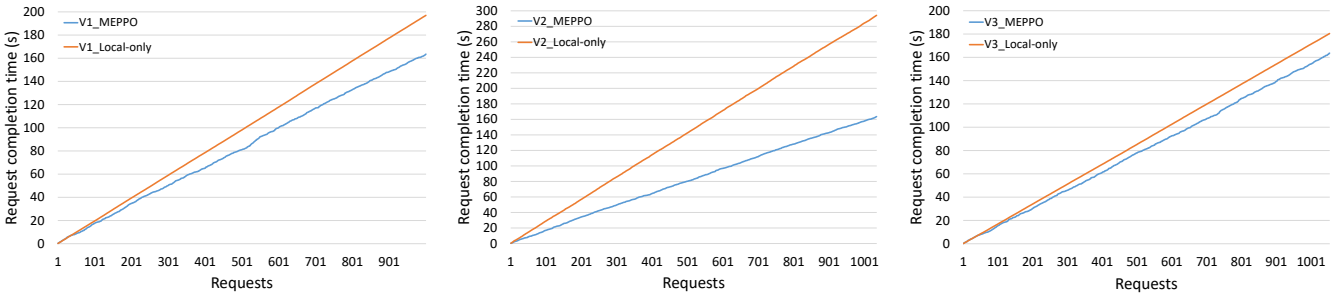
[4] J. H. Gawron, G. A. Keoleian, R. D. De Kleine, T. J. Wallington, and H. C. Kim, "Life cycle assessment of connected and automated vehicles: sensing and computing subsystem and vehicle level effects," *Environmental science & technology*, vol. 52, no. 5, pp. 3249–3256, 2018.

[5] R. Meneguette, R. De Grande, J. Ueyama, G. P. R. Filho, and E. Madeira, "Vehicular edge computing: architecture, resource management, security, and challenges," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–46, 2021.

[6] H. Gao, X. Wang, W. Wei, A. Al-Dulaimi, and Y. Xu, "Comddpg: Task offloading based on multiagent reinforcement learning for information-communication-enhanced mobile edge computing in the internet of vehicles," *IEEE Transactions on Vehicular Technology*, 2023.

[7] H. Gao, W. Huang, T. Liu, Y. Yin, and Y. Li, "Ppo2: Location privacy-oriented task offloading to edge computing using reinforcement learning for intelligent autonomous transport systems," *IEEE transactions on intelligent transportation systems*, 2022.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.



(a) Vehicle 1 (b) Vehicle 2 (c) Vehicle 3

Fig. 15: Comparison of request completion time between using the proposed MEPPO-based scheduling and local-only scheduling.

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[10] X. Xu, K. Liu, P. Dai, F. Jin, H. Ren, C. Zhan, and S. Guo, "Joint task offloading and resource optimization in noma-based vehicular edge computing: A game-theoretic drl approach," *Journal of Systems Architecture*, vol. 134, p. 102780, 2023.

[11] Y. Ju, Y. Chen, Z. Cao, L. Liu, Q. Pei, M. Xiao, K. Ota, M. Dong, and V. C. Leung, "Joint secure offloading and resource allocation for vehicular edge computing network: A multi-agent deep reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[12] S. Wang, X. Song, H. Xu, T. Song, G. Zhang, and Y. Yang, "Joint offloading decision and resource allocation in vehicular edge computing networks," *Digital Communications and Networks*, 2023.

[13] N. Waqar, S. A. Hassan, A. Mahmood, K. Dev, D.-T. Do, and M. Gidlund, "Computation offloading and resource allocation in mec-enabled integrated aerial-terrestrial vehicular networks: A reinforcement learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21 478–21 491, 2022.

[14] J. He, Y. Wang, X. Du, and Z. Lu, "V2v-based task offloading and resource allocation in vehicular edge computing networks," *arXiv preprint arXiv:2112.15065*, 2021.

[15] S. Li, X. Hu, and Y. Du, "Deep reinforcement learning for computation offloading and resource allocation in unmanned-aerial-vehicle assisted edge computing," *Sensors*, vol. 21, no. 19, p. 6499, 2021.

[16] J. Gao, Z. Kuang, J. Gao, and L. Zhao, "Joint offloading scheduling and resource allocation in vehicular edge computing: A two layer solution," *IEEE Transactions on Vehicular Technology*, 2022.

[17] W. Fan, Y. Su, J. Liu, S. Li, W. Huang, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for vehicular edge computing based on v2i and v2v modes," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[18] W. Fan, J. Liu, M. Hua, F. Wu, and Y. Liu, "Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 5314–5330, 2022.

[19] L. Liu, J. Feng, X. Mu, Q. Pei, D. Lan, and M. Xiao, "Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[20] J. Huang, J. Wan, B. Lv, Q. Ye, and Y. Chen, "Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning," *IEEE Systems Journal*, 2023.

[21] X. Li, Y. Qin, J. Huo, and W. Huangfu, "Heuristically assisted multiagent rl-based framework for computation offloading and resource allocation of mobile edge computing," *IEEE Internet of Things Journal*, 2023.

[22] G. Ma, X. Wang, M. Hu, W. Ouyang, X. Chen, and Y. Li, "Drl-based computation offloading with queue stability for vehicular-cloud-assisted mobile edge computing systems," *IEEE Transactions on Intelligent Vehicles*, 2022.

[23] P. Lang, D. Tian, X. Duan, J. Zhou, Z. Sheng, and V. C. Leung, "Blockchain-based cooperative computation offloading and secure handover in vehicular edge computing networks," *IEEE Transactions on Intelligent Vehicles*, 2023.

[24] H. Guo, J. Liu, J. Ren, and Y. Zhang, "Intelligent task offloading in vehicular edge computing networks," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 126–132, 2020.

[25] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5449–5465, 2020.

[26] S. Vemireddy and R. R. Rout, "Fuzzy reinforcement learning for energy efficient task offloading in vehicular fog computing," *Computer Networks*, vol. 199, p. 108463, 2021.

[27] C. Sonmez, C. Tunca, A. Ozgovde, and C. Ersoy, "Machine learning-based workload orchestrator for vehicular edge computing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 4, pp. 2239–2251, 2020.

[28] C. Shang, Y. Sun, H. Luo, and M. Guizani, "Computation offloading and resource allocation in noma-mec: A deep reinforcement learning approach," *IEEE Internet of Things Journal*, 2023.

[29] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 1, pp. 360–374, 2020.

[30] J. Guo, Y. Zhang, X. Chen, S. Yousefi, C. Guo, and Y. Wang, "Spatial stochastic vehicle traffic modeling for vanets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 416–425, 2017.

[31] HUAWEI, "Power and signal strength," https://support.huawei.com/enterprise/en/doc/EDOC1000113315/c3242b10/power-and-signal-strength, 2022, [Online; accessed 10-June-2023].

[32] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9637–9650, 2020.

[33] J. Feng, G. Jiang, Y. Ding, X. Zhang, H. Liu, and Y. Fan, "Antenna performance test method and result analysis for v2x communication of connected vehicle," in *Journal of Physics: Conference Series*, vol. 1607, no. 1. IOP Publishing, 2020, p. 012082.

[34] N. OpenData, "Automated traffic volume counts," https://data.cityofnewyork.us/Transportation/Automated-Traffic-Volume-Counts/7ym2-wayt, 2023, [Online; accessed 1-May-2023].

Peisong Li received a B.S. degree from the Guilin University of Electronic Technology in 2017 and a M.S. degree from the Shanghai Maritime University, China, in 2020. He is currently pursuing the Ph.D. degree with University of Liverpool, UK. His research interests include edge computing, vehicular edge computing, edge AI, and deep reinforcement learning.

This article has been accepted for publication in IEEE Transactions on Intelligent Vehicles. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TIV.2023.3321679

Ziren Xiao received a B.Sc degree in Computing and Software Systems and an M.Sc in Computer Science degree from The University of Melbourne. His past work focused on outlier detection using machine learning algorithms, and he is currently working on developing reinforcement learning approaches for transportation systems.

Xinheng Wang received the B.E. and M.Sc. degrees in electrical engineering from Xian Jiaotong University, Xi'an, China, in 1991 and 1994, respectively, and the Ph.D. degree in electrical engineering from Brunel University, Uxbridge, U.K., in 2001. He is currently a Professor with the School of Advanced Technology and was the founding Head of Department of Mechatronics and Robotics, Xi'an Jiaotong-Liverpool University (XJTLU), Suzhou 215123, China. Prior to joining XJTLU, he was a professor with different universities in the UK.

He has been an Investigator or Co-Investigator of 30+ research projects sponsored from EU, UK EPSRC, Innovate UK, China NSFC, and industry. He has authored or coauthored 220+ referred papers. He holds 22 granted patents, including 1 US, 1 Japan, 4 South Korea and 16 China patents. He was one of the key developers of the world's first smart trolley to provide intelligent services to passengers at airports. He is currently leading the XJTLU-uGo Robotics Research Centre with multi-million Yuan sponsorship from industry to develop airport robots. His current research interests include intelligent and connected systems, including robotics and healthcare systems, indoor acoustic localization, acoustic posture detection and recognition, digitalization of traditional Chinese medicine, and SLAM and navigation for robots.

Kaizhu Huang works on machine learning, neural information processing, and pattern recognition. He holds tenured Full Professorship of ECE and directors Data Science Research Center at Duke Kunshan University (DKU). Prof. Huang obtained his PhD degree from Chinese University of Hong Kong (CUHK) in 2004, Master degree from Institute of Automation, Chinese Academy of Sciences in 2000, and Bachelor degree from Xi'an Jiaotong-Liverpool University in 1997. He worked in Fujitsu Research Centre, CUHK, University of Bristol, National Laboratory of Pattern Recognition, Chinese Academy of Sciences, Xi'an Jiaotong-Liverpool University from 2004 to 2021. He was the recipient of 2011 Asia Pacific Neural Network Society Young Researcher Award. He received best (runner-up) paper or book award eight times in major international/ national conferences. He serves as associated editors/advisory board members in six international journals and book series (e.g. Pattern Recognition Journal, and Neural Network Journal). He was invited as a keynote/tutorial speaker in more than 40 international conferences or workshops.

Yi Huang received BSc in Physics (Wuhan, China) in 1984, MSc (Eng) in Microwave Engineering (Nanjing, China) in 1987, and DPhil in Communications from the University of Oxford, UK in 1994. He has been conducting research in the areas of antennas, wireless communications, applied electromagnetics, radar, and EMC since 1987. More recently, he has focused on new materials for antennas, wireless energy harvesting, and power transfer. His experience includes 3 years spent with NRIET (China) as a Radar Engineer and various periods with the Universities of Birmingham, Oxford, and Essex in the UK as a member of research staff. He worked as a Research Fellow at British Telecom Labs in 1994 and then joined the Department of Electrical Engineering & Electronics, the University of Liverpool, UK as a Faculty in 1995, where he is now a full Professor in Wireless Engineering, the Head of High-Frequency Engineering Group.

Prof Huang has published over 500 refereed papers in leading international journals and conference proceedings and authored four books, including Antennas: from Theory to Practice (John Wiley, 2008 and 2021). He has received many patents, and research grants from research councils, government agencies, charities, the EU, and industry, and is a recipient of over 10 awards (e.g. EuCAP2023 Best Antenna Paper, IET Premium Award for Best Papers 2022, IET Innovation Award 2018, and BAE Systems Chairman's Award 2017). He has served on a number of national and international technical committees and has been an Editor, Associate Editor, or Guest Editor of seven international journals (including IEEE TAP 2022-date, IEEE AWPL 2016-2022). In addition, he has been a keynote/invited speaker and organiser of many conferences and workshops (e.g. IEEE iWAT2010, LAPC2012, UCMMT2017/2023, and EuCAP2018/2024). He is at present the Editor-in-Chief of Wireless Engineering and Technology, the UK and Ireland Rep to the European Association of Antenna and Propagation (EurAAP, 2016-2020, 2022-date), a Fellow of IET, a Fellow of IEEE, and a Distinguished Lecturer of IEEE AP-S (2022-2025). More information can be found from: https://www.liverpool.ac.uk/electrical-engineering-and-electronics/staff/yi-huang/

Honghao Gao is currently with the School of Computer Engineering and Science, Shanghai University, China. He is also a Professor at the College of Future Industry, Gachon University, Korea. Prior to that, he was a Research Fellow with the Software Engineering Information Technology Institute at Central Michigan University, USA, and was an Adjunct Professor at Hangzhou Dianzi University, China. His research interests include Software Intelligence, Cloud/Edge Computing, and AI4Healthcare. He has publications in IEEE TII, IEEE T-ITS, IEEE TNNLS, IEEE TMM, IEEE TSC, IEEE TCC, IEEE TFS, IEEE TNSE, IEEE TNSM, IEEE TCCN, IEEE TGCN, IEEE TCSS, IEEE TETCI, IEEE TCE, IEEE/ACM TCBB, etc. He was the 2022 recipient of Highly Cited Chinese Researchers by Elsevier, and the 2021 recipient of IEEE Outstanding Paper Award for the IEEE Transactions on Industrial Informatics.

Prof. Gao is a Fellow of the Institution of Engineering and Technology (IET), a Fellow of the British Computer Society (BCS), and a Member of the European Academy of Sciences and Arts (EASA). He is the Editor-in-Chief for International Journal of Web Information Systems (IJWIS), Editor for Wireless Network (WINE), The Computer Journal (COMPJ), and IET Wireless Sensor Systems (IET WSS), and Associate Editor for IEEE Transactions on Intelligent Transportation Systems (IEEE T-ITS), IET Intelligent Transport Systems (IET ITS), IET Software, International Journal of Communication Systems (IJCS), Journal of Internet Technology (JIT), and Engineering Reports (EngReports). Moreover, he has broad working experience in cooperative industry-university-research. He is a European Union Institutions-appointed external expert for reviewing and monitoring EU Project, is a member of the EPSRC Peer Review Associate College for UK Research and Innovation in the UK, and a founding member of the IEEE Computer Society Smart Manufacturing Standards Committee.